

# The Curse of Correlation in Security Games and Principle of Max-Entropy

Haifeng Xu Milind Tambe Shaddin Dughmi Venil Loyd Noronha

University of Southern California, Los Angeles, USA

{haifengx, tambe, shaddin, vnoronha}@usc.edu

## Abstract

In this paper, we identify and study a fundamental, yet underexplored, phenomenon in security games, which we term the *Curse of Correlation* (CoC). Specifically, we observe that there is *inevitable* correlation among the protection status of different targets. Such correlation is a crucial concern, especially in spatio-temporal domains like conservation area patrolling, where attackers can monitor patrollers at certain areas and then infer their patrolling routes using such correlation. To mitigate this issue, we introduce the *principle of max-entropy* to security games, and focus on designing entropy-maximizing defending strategies for the spatio-temporal security game – a major victim of CoC. We prove that the problem is #P-hard in general, but propose efficient algorithms in well-motivated special settings. Our experiments show significant advantages of the max-entropy algorithms against previous algorithms.

## 1 Introduction

The security game, played between a *defender* and an *attacker*, concerns randomized allocation of limited *security resources* to protect *targets* [Basilico *et al.*, 2009a; Tambe, 2011; Blum *et al.*, 2014; Yin *et al.*, 2015]. Standard models assume that the attacker cannot observe any information about the defender’s real-time deployment. However, this assumption may fail since in some settings the attacker can *partially* observe the defender’s real-time strategy. Such partial observation can be utilized to infer more information about the defending strategy. For example, it has been reported that in conservation enforcement, some poachers partially monitor the rangers’ patrolling activity and then base their poaching on these observations [Nyirenda and Chomba, 2012; Moreto, 2013]. Similar concerns are raised when designing defending strategies for patrolling on graphs [Agmon *et al.*, AAMAS 2008; Basilico *et al.*, 2009a], patrolling airports [Xu *et al.*, 2015] and scheduling of air marshals [Mail, Nov 11 2016]. Though previously work has addressed some aspects of this issue mainly from a game-theoretic perspective, in this paper we propose a systematic way to analyze and address this phenomenon from a new information-theory perspective.

We observe that the randomized allocation of limited security resources creates *inherent* correlation in the protection of targets – that some targets get protected must mean that other targets are not protected. Such correlation allows the attacker to utilize his partial observation of some targets to infer the protection status of other targets, a phenomenon which we term the *Curse of Correlation* (CoC). To formally model this phenomenon, we introduce a novel notion named the Total Deviation of Marginals (TDM) to quantify the uncertainty loss of a random patrolling strategy due to the attacker’s partial observations. Unfortunately, we formally prove that such uncertainty loss is *inevitable*, and the more the attacker observes, the higher the loss is. To mitigate CoC, we introduce the *principle of max-entropy* [Jaynes, 1957] to security games, and propose to adopt the strategy that, subject to optimizing the usual objective under necessary constraints, maximizes entropy. The underlying intuition is that a patrolling strategy with high entropy has less correlation among targets, therefore is more “robust” to the attacker’s partial observations. We then move to our technical contributions of designing algorithms to compute defending strategies with maximum or high entropy. We prove that the exact max-entropy defending strategy is #P-hard to compute in general, but admits efficient and practical algorithms in well-motivated special cases as well as well-performed heuristic algorithms. Our experiments show significant advantages of using entropy-maximizing algorithms over classic algorithms in presence of partial attacker observations, illustrating the necessity of handling CoC. The experiments also justify how the concerns raised at the beginning of this section can be mitigated.

We note that previous works on adversarial patrolling games (APGs) have also considered attackers’ monitoring of the defender’s real-time strategies [Agmon *et al.*, AAMAS 2008; Agmon *et al.*, ICRA 2008; Basilico *et al.*, 2009a; Basilico *et al.*, 2009b; Bošanský *et al.*, 2011; Vorobeychik *et al.*, 2014]. However the models considered in these works differ from ours. First, their models usually assume a particular attacker model (e.g., many of these works assume that the attacker can monitor all targets at current time), which we do not require in this paper. Second, the APG game is always played out in space and time, and the attacker in the game takes time to complete an attack, while these assumptions are not required in our analysis and algorithms (though we do consider games played out in space and time). Moreover, our

primary focus in this paper is to formulate, analyze and mitigate the correlation among targets, which has not been examined before to our knowledge. [Xu *et al.*, 2015] does consider entropy maximizing in a simple security game setting, however their algorithm does *not* handle the case where there are resource scheduling constraints.

## 2 Preliminaries

The security game is played between a defender and an attacker. The defender aims to use limited security resources to protect  $n$  targets, denoted by set  $[n]$ , from the attacker’s attack. A *defender pure strategy*  $S \subseteq [n]$  is a subset of targets that can be feasibly covered by the security resources. A more convenient representation of  $S$  is a binary vector  $\mathbf{s} \in \{0, 1\}^n$ , indicating whether each target is covered or not. We will use  $S$  or  $\mathbf{s}$  interchangeably throughout, and the meaning should be clear from context. We use  $\mathcal{S}$  to denote the set of all defender pure strategies. A *defender mixed strategy* is a distribution  $\mathbf{p}$  over  $\mathcal{S}$ . We note that the size of  $\mathcal{S}$  is usually *exponentially* large in the game representation [Xu, 2016].

We do not assume a particular attacker behavior model, and allow the attacker to be rational or irrational, attack one or multiple targets. For the purpose of this paper, the concrete utility structure of the game is not particularly important *except* that we require players’ *expected utilities* to depend only on the *marginal probability* that each target is covered. This is assumed by most security games, even with rationality or uncertainty concerns [Nguyen *et al.*, 2014].

**Marginal Probabilities and Implementability.** For any mixed strategy  $\mathbf{p}$ ,  $x_i = \sum_{\mathbf{s} \in \mathcal{S}} p_{\mathbf{s}} \cdot s_i \in [0, 1]$  is the *marginal (coverage) probability* of target  $i$ . Let vector  $\mathbf{x} = (x_1, \dots, x_n)^T = \sum_{\mathbf{s} \in \mathcal{S}} p_{\mathbf{s}} \cdot \mathbf{s} \in [0, 1]^n$  be the *marginal vector* of all targets. In this case, we say mixed strategy  $\mathbf{p}$  *implements* marginal vector  $\mathbf{x}$ , and  $\mathbf{x}$  is *implementable*. Note that not any  $\mathbf{x} \in [0, 1]^n$  is implementable – all the implementable  $\mathbf{x}$ ’s form a polytope  $\text{conv}(\mathcal{S})$ , i.e., the convex hull of  $\mathcal{S}$ .

There are generally many mixed strategies that implement the same desired marginal vector, and most algorithms for security games are designed to compute a mixed strategy with *small support*. One primary goal of this paper is to understand which of these mixed strategies is the best. In fact, we assume throughout that *the optimal marginal vector is given to us, thus we can restrict our attention to computing the best mixed strategy that implements it*. Next, we discuss a particular way to implement any given marginal vector  $\mathbf{x}$ .

**The Max-Entropy Implementation.** Given any implementable  $\mathbf{x}$ , the *max-entropy implementation* of  $\mathbf{x}$  is the mixed strategy  $\mathbf{p}$  of maximum (Shannon) entropy among all mixed strategies that implement  $\mathbf{x}$ . More precisely, the max-entropy implementation of  $\mathbf{x}$  is the optimal solution to the following convex program (CP) with variable  $\theta_{\mathbf{s}}$ ’s.

$$\begin{aligned} & \text{maximize} && -\sum_{\mathbf{s} \in \mathcal{S}} \theta_{\mathbf{s}} \log(\theta_{\mathbf{s}}) \\ & \text{subject to} && \sum_{\mathbf{s}: i \in \mathbf{s}} \theta_{\mathbf{s}} = x_i, && \text{for } i \in [n]. \\ & && \sum_{\mathbf{s} \in \mathcal{S}} \theta_{\mathbf{s}} = 1 \\ & && \theta_{\mathbf{s}} \geq 0, && \text{for } \mathbf{s} \in \mathcal{S}. \end{aligned} \quad (1)$$

An algorithm solves CP (1) if it takes  $\mathbf{x}$  as input and randomly samples  $S \in \mathcal{S}$  with probability  $\theta_{\mathbf{s}}^*$  where  $\theta^*$  is the

optimal solution to CP (1). An obvious challenge of solving CP (1) is that the optimal  $\theta^*$  usually has exponentially large support [Singh and Vishnoi, 2014]. It turns out that this can be overcome via algorithms that can efficiently sample a set  $S$  “on the fly” [Xu *et al.*, 2015]. In fact, we can solve CP (1) efficiently so long as we can solve the following *generalized counting* problem over  $\mathcal{S}$ .

**Definition 2.1** (Generalized Counting). *Given any  $\alpha \in R_+^n$ , compute  $C(\alpha) = \sum_{S \in \mathcal{S}} \alpha_S$ , where  $\alpha_S = \prod_{i \in S} \alpha_i$ .*

Note that  $C(\mathbf{1})$  equals precisely the cardinality of  $\mathcal{S}$ . More generally,  $C(\alpha)$  counts the total elements in  $\mathcal{S}$  by weighting each element  $S$  with  $\alpha_S = \prod_{i \in S} \alpha_i$ .

**Lemma 2.2.** [Singh and Vishnoi, 2014] *The max-entropy implementation of  $\mathbf{x}$  over  $\mathcal{S}$  can be computed in polynomial time<sup>1</sup> for any implementable  $\mathbf{x}$  if and only if the generalized counting problem over  $\mathcal{S}$  can be solved in polynomial time.*

As a result of Lemma 2.2, to compute the max-entropy implementation of any marginal vector  $\mathbf{x}$  over  $\mathcal{S}$ , we only need to design a generalized counting algorithm for  $\mathcal{S}$ . However, we do remark that this does *not* magically simplify the problem since the generalized counting problem is computationally hard in most cases. In fact, *very few* efficient algorithms are known, even for counting simple combinatorial structures like perfect bipartite matchings [Valiant, 1979].

## 3 The Curse of Correlation and Principle of Maximum Entropy

The essential charge of security games, regardless of which model or algorithm adopted, is to create *unpredictability via randomization*. Most of previous research has built on the critical assumption that the attacker cannot observe any information regarding the defender’s real-time deployment. However, this assumption may fail in practice since in many cases the attacker can obtain *partial observations* [Moreto, 2013; Basilico *et al.*, 2009a; Xu *et al.*, 2015]. Due to the correlation among targets, the attacker could infer information about the protection status of targets from their partial observations. We illustrate this via a concrete example.

**Hacking Security Games with One Bit of Information.** Our example is described in the setting of conservation area protection, though we emphasize that the general phenomenon it illustrates occurs in many security games. The problem concerns designing rangers’ patrolling paths through a fixed time period, e.g., a day. This is usually modeled by discretizing the area to be patrolled into cells as well as the time. At the top of Figure 1, we depicts a concrete example with 4 cells to be protected at 3 time points: morning, noon and afternoon. The numbers around each cell is the required marginal probabilities for each cell at each time (color thickness corresponds to the probabilities). The defender has 2 rangers, and wants to randomize their patrolling to achieve the marginal probabilities listed in the figure.

If one insists on mixed strategies with small support as traditional algorithms do, the marginal vector can be im-

<sup>1</sup>By “computed” we mean the problem is solved within machine precision, since computers cannot solve convex programs exactly.

plemented by mixing the three pure strategies listed at the bottom of Figure 1 (filled dots are fully covered).

Unfortunately, it turns out that such an implementation is extremely vulnerable to the attacker's partial observation. For example, if the attacker can monitor the status of the top left cell in the morning (i.e., the one with dashed boundary) and prepare an attack in the afternoon, he can always find a completely uncovered cell to attack. In particular, if the dashed cell is covered, this means Strategy 3 is deployed, and two cells are uncovered in the afternoon; Otherwise, either Strategy 1 or 2 is deployed, and the bottom left cell is uncovered in the afternoon for sure. So the attacker successfully identifies uncovered cells in the afternoon by monitoring only *one* cell in the morning.

Concerns like these motivate our formal treatment about effects of the attacker's partial observation and how to deal with it. Unfortunately, we prove that such partial observation of certain targets indeed *inevitably degrades the unpredictability* of other unobserved targets (in security games, the more unpredictable, the better). We start with a novel notion to quantify the unpredictability loss in security games.

### 3.1 Quantifying The Loss of Unpredictability

To formally examine the issue, we introduce a natural notion of mathematically quantify the loss of unpredictability in security games. We begin with some notations. Recall that any mixed strategy  $\mathbf{p}$  is a distribution over  $\mathcal{S}$ . Here, we require a slightly different view, and will instead treat a mixed strategy as a random binary vector  $X = (X_1, \dots, X_n) \in \{0, 1\}^n$  satisfying  $\Pr(X = \mathbf{s}) = p_{\mathbf{s}}$ . Note that  $X_i \in \{0, 1\}$  is the random protection status (i.e., covered or not) of target  $i$ , and  $\Pr(X_i = 1) = x_i$  is its marginal coverage probability. For any subset  $T \subseteq [n]$ , we use  $X_T$  to denote the set  $\{X_i\}_{i \in T}$ , i.e., the protection status of targets in  $T$ . As a particular case,  $X_{-i} = \{X_k : k \neq i\}$  contains the status of all targets except  $i$ .  $v_T, v_{-i}$  are defined similarly for any vector  $v \in \mathbb{R}^n$ . Note that,  $X_i$ 's are usually correlated. We will constantly talk about conditional probabilities. We use  $X_i(v_A)$  to denote the random variable  $X_i$  conditioned on  $X_A = v_A$  for some  $v_A \in \{0, 1\}^A$ , and  $x_i(v_A) = \Pr(X_i(v_A) = 1) = \Pr(X_i = 1 | X_A = v_A)$  is the marginal probability of target  $i$  conditioned on  $X_A = v_A$ . We view  $x_i(X_A)$  as a random variable with randomness inherited from  $X_A$ , i.e.,  $\Pr[x_i(X_A) = x_i(v_A)] = \Pr(X_A = v_A)$ .

To illustrate, consider a simple example with two targets  $\{1, 2\}$  and the defender picks a target uniformly at random to protect. Then  $x_2(X_1)$  is a random variable with  $\Pr[x_2(X_1) = 1] = \Pr[X_1 = 0] = \frac{1}{2}$  and  $\Pr[x_2(X_1) = 0] = \Pr[X_1 = 1] = \frac{1}{2}$ .

We consider the following natural concept termed the **Total Deviation of Marginals (TDM)**. Formally, given any mixed

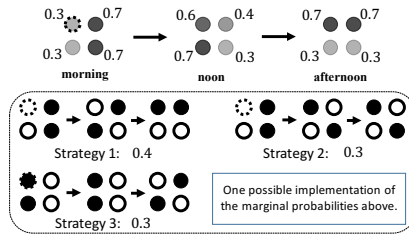


Figure 1: Marginal and Implementation.

strategy  $X$ , for any  $T, A \subseteq [n]$ , TDM of  $X_T$  conditioned on  $X_A$ , is defined as follows:

$$\text{TDM}(X_T | X_A) = \sum_{i \in T} \sqrt{\mathbf{E}_{v_A \sim X_A} [x_i - x_i(v_A)]^2},$$

where  $v_A \sim X_A$  means  $v_A$  follows the distribution of  $X_A$ .

Note that  $\sqrt{\mathbf{E}_{v_A \sim X_A} [x_i - x_i(v_A)]^2}$  is precisely the standard deviation of  $x_i(X_A)$ . Thus  $\text{TDM}(X_T | X_A)$  is the sum of the standard deviations of  $x_i(X_A)$  for  $i \in T$ . In the above example,  $\text{TDM}(X_2 | X_1) = \sqrt{(0.5 - 1)^2/2 + (0.5 - 0)^2/2}$ . Note that  $T$  and  $A$  are allowed to intersect. We interpret  $T$  as the target set of the attacker's interest and  $A$  as the set of targets that the attacker can observe. For example, in wildlife protection domain,  $T$  could be the set of areas with animals and  $A$  could be the set of the attacker's neighborhood areas that can be monitored.

We argue that TDM is a natural notion to capture the unpredictability loss of targets in set  $T$  when the attacker can observe targets in set  $A$ . Standard deviation has been widely used to characterize unpredictability (or uncertainty) of random variables [Bernardo and Smith, 2001]. By observing the status of targets in set  $A$ , the attacker observes realizations of the random variable  $x_i(X_A)$  for any  $i \in T$ , thus the uncertainty loss from targets in set  $T$  captured by their total deviation is precisely the TDM as we defined.<sup>2</sup>

### 3.2 The Curse of Correlation

Note that TDM is the loss of unpredictability, so a small value is preferred. For any mixed strategy  $X$  and any  $i \in [n]$ , we say target  $i$  is *trivial* in  $X$  if  $\Pr(X_i = 1) = 0$  or  $1$ .<sup>3</sup> Otherwise, we say  $i$  is non-trivial. Obviously, if a trivial target is monitored, the attacker infers no information about other targets. The following theorem shows that if a non-trivial target is monitored by the attacker, the unpredictability loss of other targets is *strictly* positive.

**Theorem 3.1. [The Curse of Correlation]** *Let  $X$  be any mixed strategy. For any sets  $A, B$  satisfying  $B \subset A \subseteq [n]$  and any  $T \subseteq [n]$ , we have*

$$\text{TDM}(X_T | X_B) \leq \text{TDM}(X_T | X_A).$$

Moreover, for any non-trivial target  $i$ ,  $\text{TDM}(X_{-i} | X_i) > 0$ .

Due to space limit, all proofs in this paper are either described with sketches or omitted. Formal proofs can be found in the online appendix.<sup>4</sup> Notice that inequality  $\text{TDM}(X_{-i} | X_i) > 0$  is strict for any non-trivial target  $i$ . This means that in *any* mixed strategy, the attacker can always infer more information about other targets by monitoring a non-trivial target. Moreover, the more targets he monitors, the more information he obtains. This illustrates the Curse of Correlation (CoC).

**Dilemma of Traditional Algorithms.** Traditional security game algorithms, e.g., comb sampling [Tsai *et al.*, 2010],

<sup>2</sup>We note that entropy could be another notion of unpredictability, and all our results in Section 3 carry over similarly for entropy.

<sup>3</sup>We assume all resources are fully used, and do not consider the (unreasonable) situations in which security resources are underused.

<sup>4</sup><http://bit.ly/21kqxbq>

strategy/column generation [Jain *et al.*, 2010], are designed to generate strategies of *small support* due to the concern of computational cost. Unfortunately, as illustrated by the previous example, such small-support strategies are very likely to induce high correlation among targets, thus exacerbates CoC.

### 3.3 The Principle of Maximum Entropy

The well-known principle of maximum entropy provides a general guide for selecting distributions. It states that subject to any prior requirements (e.g., data, constraints, etc.), the probability distribution which best represents the current state is the one with maximum entropy. In security games, we constantly face the question of computing a defender mixed strategy – i.e., a distribution over pure strategies – subject to scheduling constraints. A carelessly chosen mixed strategy is likely to have high correlation among targets, and suffers severely from CoC. To mitigate this issue, it is important to select a mixed strategy with less correlation. The most natural choice here is perhaps the distribution with largest entropy. We thus propose the following principle in security games.

**The Principle of Maximum Entropy in Security Games.** *Consider a security game with any objective and scheduling constraints. Among all the mixed strategies that optimize the objective subject to scheduling constraints, the best mixed strategy is the one with maximum entropy.*

The underlying intuition of the principle is that the max-entropy distribution is the most uninformative distribution, thus is most unpredictable by attackers. Revisiting the example at the beginning of this section, if we let  $T$  be the set of all the four cells in the afternoon and  $A$  be the top left cell in the morning, it can be verified that  $\text{TDM}(X_T|X_A) = 1.31$  in the simple mixed strategy in Figure 1, but  $\text{TDM}(X_T|X_A) = 0.17$  in the max-entropy mixed strategy that implements the given marginal vector. This big gap illustrates the power of the principle in preserving unpredictability.

## 4 Efficient Max-Entropy Implementation for Spatio-Temporal Security Games

In many applications the security game is played out in space and time [Basilico *et al.*, 2009a; Fang *et al.*, 2013; Yin *et al.*, 2015], which is termed the *spatio-temporal* security game. In these domains, the defender needs to move patrollers as time goes on since the targets dynamically move or change. Due to the inherent correlation among the patroller’s consecutive moves, these games easily fall victim to CoC. In this section, we seek to mitigate CoC using the principle of maximum entropy. In particular, given any optimal marginal vector (obtainable from any algorithm solving the original game), we seek to compute its max-entropy implementation.

Like most of the previous work (e.g., all the cited ones above), we focus on discretized spatio-temporal security games. Such a game is described by a  $T \times N$  grid graph  $G = (V, E)$  indicating a problem with  $N$  cells and  $T$  time layers (see Figure 2). We will use  $v_{t,i}$  to denote a grid node, representing cell  $i$  at time  $t$ . Each  $v_{t,i}$  is treated as a target, so there are  $n = T \times N$  targets. The directed edges between each consecutive time layers denote

the patroller’s feasible moves between cells within consecutive time layers. Such feasibility usually incorporates speed limit, terrain and other real-field constraints. Figure 2 depicts some feasible moves between time layer 1 and 2.

A feasible patrol path is a path in  $G$  starting from time 1 and ends at time  $T$  (e.g., the dashed path in Figure 2). Note that there are exponentially many patrol paths. We assume the defender has  $k$  homogeneous patrollers, therefore a defender pure strategy corresponds to the *set of nodes* covered by  $k$  feasible patrol paths. We start by proving

that the max-entropy implementation problem (see Section 2) for spatio-temporal security games is computationally hard.

**Theorem 4.1.** *It is #P-hard to compute the max-entropy implementation for spatio-temporal security games even when there are two time layers.*

Theorem 4.1 suggests that there is unlikely an efficient algorithm for the max-entropy implementation problem in general. Moreover, there is evidence that the generalized counting problem does not even admit an optimization formulation (e.g., integer or non-convex program) [Toda, 1991], therefore we cannot utilize state-of-the-art optimization software to tackle the problem. Nevertheless, we show that the max-entropy implementation problem admits efficient and practical algorithms in two well-motivated special cases.

### 4.1 Case 1: Small Number of Patrollers

In many applications, the defender only possesses a small number of patrollers. For example, in wildlife protection, the defender usually has only one or two patrol teams at each patrol post [Fang *et al.*, 2016]; the US Coast Guard uses two patrollers to protect Staten Island ferries [Fang *et al.*, 2013]. In this section, we show that when the number of patrollers is small (i.e., can be treated as a constant), the max entropy implementation can be computed efficiently.

**Theorem 4.2.** *When the number of patrollers is a constant, the max-entropy implementation for any spatio-temporal security game can be computed in  $\text{poly}(N, T)$  time.*

We prove Theorem 4.2 by developing an efficient algorithm for the generalized counting version of the problem (which is all we need to generate an algorithm by Lemma 2.2). For ease of presentation, our description focuses on the case with two patrollers, though it generalizes to a constant number of patrollers. We propose a dynamic program (DP) that exploits the natural chronological order of the targets along the temporal dimension. In particular, given any set of weights  $\{\alpha_{t,i}\}$  for the generalized counting problem, for any  $1 \leq i \leq j \leq N$ , we use  $\text{DP}(i, j; t)$  to denote the solution to the counting problem restricted to the truncated

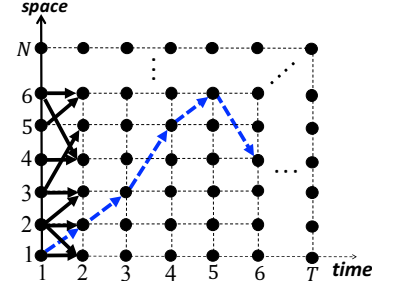


Figure 2: Structure of a Spatio-Temporal Security Game

graph with only time layers  $1, 2, \dots, t$ , and moreover, satisfying that the two patroller must end at cell  $i, j$  at time  $t$ . Note that  $\text{DP}(i, j; 1) = \alpha_{1,i} \alpha_{1,j}$ . We then use the following update rule for  $t \geq 2$ :

$$\text{DP}(i, j; t) = \alpha_{t,i} \alpha_{t,j} \cdot \sum_{(i', j') \in \text{pre}(i, j)} \text{DP}(i', j'; t-1)$$

where  $\text{pre} = \{(i', j') : i' \leq j' \text{ s.t. } v_{t-1,i'}, v_{t-1,j'} \text{ can reach } v_{t,i}, v_{t,j}\}$  is essentially the set of all pairs of nodes that can reach  $v_{t,i}, v_{t,j}$ . Note that the solution to the generalized counting problem is  $\sum_{i \leq j} \text{DP}(i, j; T)$ .

## 4.2 Case 2: Air Marshal Scheduling for Protecting Round-Trip Flights

One important task faced by the Federal Air Marshal Service (FAMS) is to schedule air marshals to protect *international* flights. In this case, the schedule of each air marshal is a *round trip* [Kiekintveld *et al.*, 2009], and the game can be described as a spatio-temporal security game with two time layers. In this section, we show that the max-entropy implementation can be computed efficiently in this setting, *despite* the fact that the #P-hardness in Theorem 4.1 holds for the case of two time layers. Our algorithm crucially explores certain “order” structure of the air marshal’s schedules.

We start by illustrating the problem. We seek to allocate  $k$  homogeneous air marshals to protect round-trip international flights originating from certain domestic city but possibly to different outside cities. These round-trip flights constitute a bipartite graph  $G = (A \cup B, E)$  in which nodes in  $A [B]$  correspond to all originating [return] flights;  $e = (A_i, B_j) \in E$  iff  $e$  forms a consistent round trip for an air marshal. The following are natural constraints on the structure of  $G$  (which is what makes the problem more tractable than general settings):  $(A_i, B_j) \in E$  if (a) the destination of  $A_i$  is the departure city of  $B_j$ ; (2) the arrival time of  $A_i$ , denoted as  $\text{arrival}(A_i)$ , and the departure time of  $B_j$ , denoted as  $\text{depart}(B_j)$ , satisfy  $\text{depart}(B_j) - \text{arrival}(A_i) \in [T_1, T_2]$  for some global constants  $T_2 > T_1 > 0$ .

**Theorem 4.3.** *The max-entropy implementation for the air marshal scheduling problem with round trips can be computed in  $\text{poly}(n, k)$  time, where  $n = |A \cup B|$  is the number of total flights and  $k$  is the number of air marshals.*

We prove Theorem 4.3 by developing an efficient algorithm for generalized counting version of the problem. Our algorithm crucially exploits the following “order” structure.

**Definition 4.4.** [Ordered Matching] In a bipartite graph  $G = (A \cup B, E)$ , a matching  $M = \{e_1, \dots, e_k\}$  is called an ordered matching if any edges  $e, e' \in M$ , with  $e = (A_i, B_j)$  and  $e' = (A_{i'}, B_{j'})$ , satisfy either  $i > i', j > j'$  or  $i < i', j < j'$ .

Visually, any two edges  $e, e'$  in an ordered matching satisfy that  $e$  is either “above” or “below”  $e'$  – they do not cross. The following lemma plays a key role in our counting algorithm.

**Lemma 4.5.** *In the air marshal scheduling problem, there is a way to order flights in  $A$  and  $B$  so that pure strategies and size- $k$  ordered matchings are in one-to-one correspondence.*

Lemma 4.5 implies that the generalized counting of pure strategies is equivalent to that of size- $k$  ordered matching. Interestingly, though counting bipartite matchings is #P-hard

[Valiant, 1979], it turns out that the generalized counting of size- $k$  ordered matchings can be solved in polynomial time via a dynamic program in a similar manner as the one in Section 4.1. We omit repetitive details here.

## 5 Randomized Carathéodory – An Efficient Heuristic Sampling Approach

In this section, we propose a heuristic sampling algorithm to implement any given implementable marginal vector  $\mathbf{x}$ . This algorithm works for *general* security games (not only spatio-temporal ones), and is *computationally efficient* so long as the underlying security can be solved efficiently.

At a high level, our idea is to implement a *randomized* algorithm for the celebrated Carathéodory’s theorem, which is about the following *existence* statement: for any bounded polytope  $\mathcal{P} \subseteq \mathbb{R}^n$  and any  $\mathbf{x} \in \mathcal{P}$ , there exists at most  $n + 1$  vertices of the polytope  $\mathcal{P}$  such that  $\mathbf{x}$  can be written as a convex combination of these vertices. Interpreting  $\mathcal{P}$  as the convex hull of defender pure strategies, this means any defender mixed strategy, i.e., a point in  $\mathcal{P}$ , can be decomposed as a mixed strategy that mixes over at most  $n + 1$  pure strategies ( $n$  is the number of targets). We will turn this existence statement into an efficient randomized algorithm, named **CARATHÉODORY RANDOMIZED DECOMPOSITION (CARD)**.

Consider any polytope  $\mathcal{P} = \{\mathbf{z} : A\mathbf{z} \leq \mathbf{b}; M\mathbf{z} = \mathbf{c}\}$  explicitly represented by polynomially many linear constraints, and any  $\mathbf{x} \in \mathcal{P}$ . We use  $A_i, b_i$  to denote the  $i$ ’th row of  $A$  and  $b$  respectively;  $A_i \mathbf{z} = b_i$  is a *facet* of  $\mathcal{P}$ . A concrete description of CARD is presented in Algorithm 1. Geometrically, CARD randomly picks any vertex  $\mathbf{v}_1$  of  $\mathcal{P}$ , then “walks along” the ray that originates from  $\mathbf{v}_1$  and points to  $\mathbf{x}$ , until it crosses a facet of  $\mathcal{P}$ , denoted by  $A_i \mathbf{z} = b_i$ , at some point  $\mathbf{v}_2$ . Thus,  $\mathbf{x}$  can be decomposed as a convex combination of  $\mathbf{v}_1, \mathbf{v}_2$ . CARD then treats  $\mathbf{v}_2$  as a new  $\mathbf{x}$  and recursively decompose it until  $\mathbf{v}_2$  becomes a vertex. Crucially, the algorithm is *randomized* since the vertex  $\mathbf{v}_1$  and all vertices during later recursions are chosen randomly.

---

### Algorithm 1 CARD

---

**Input:**  $\mathcal{P} = \{\mathbf{z} \in \mathbb{R}^n : A\mathbf{z} \leq \mathbf{b}; M\mathbf{z} = \mathbf{c}\}$  and  $\mathbf{x} \in \mathcal{P}$

**Output:**  $\mathbf{v}_1, \dots, \mathbf{v}_k$  and  $p_1, \dots, p_k$  such that  $\sum \mathbf{v}_i p_i = \mathbf{x}$ .

---

- 1: **if**  $\text{rank}(M) = n$  **then**
  - 2:   Return the unique point  $\mathbf{v}_1$  in  $\mathcal{P}$  and  $p_1 = 1$ .
  - 3: **else**
  - 4:   Choose  $\mathbf{a} \in [-1, 1]^n$  uniformly at random
  - 5:   Compute  $\mathbf{v}_1 = \arg \max_{\mathbf{z} \in \mathcal{P}} \langle \mathbf{a}, \mathbf{z} \rangle$ .
  - 6:   Compute  $t = \min_{i: A_i(\mathbf{x} - \mathbf{v}_1) > 0} \frac{b_i - A_i \mathbf{x}_i}{A_i(\mathbf{x} - \mathbf{v}_1)}$ ; Let  $i^*$  be the row obtaining  $t$ , and  $\mathcal{P}' = \{\mathbf{z} \in \mathcal{P} : A_{i^*} \mathbf{z} = b_{i^*}\}$ .
  - 7:    $\mathbf{v}_2 = \mathbf{x} + t(\mathbf{x} - \mathbf{v}_1)$ ;  $p_1 = \frac{t}{t+1}$ ,  $p_2 = \frac{1}{1+t}$ .
  - 8:    $[V', \mathbf{p}'] = \text{CARD}(\mathbf{v}_2, \mathcal{P}')$
  - 9:   **return**  $V = (\mathbf{v}_1, V')$  and  $\mathbf{p} = (p_1, p_2 \times \mathbf{p}')$ .
  - 10: **end if**
- 

## 6 Experimental Evaluations

The most widely used approach for solving large-scale security games is the *column generation* technique (a.k.a., strat-

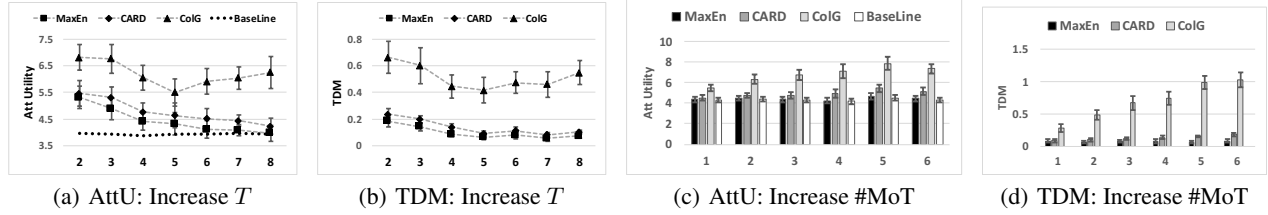


Figure 3: Comparisons in the WildProt Setting:  $N = 9$ , and  $T = 9$  in (c) and (d); #MoT = number of monitored targets.

egy/constraint generation [Jain *et al.*, 2010; Bosansky *et al.*, 2015]). We now experimentally compare our algorithms with traditional column-generation-based algorithms in settings with partial attacker observations. In particular, we compare the following three algorithms: 1. **MaxEn**: the max entropy implementation of the optimal marginal vector; 2. **CARD**: CARathéodory Randomized Decomposition (CARD) of the optimal marginal vector; 3. **ColG**: optimal mixed strategy computed by the column generation technique.

We remark that **ColG** instantiated in different settings corresponds to different concrete algorithms. Here, however, we call all these algorithms **ColG** to emphasize the underlying algorithmic basics, i.e., the column generation technique, and our goal is to examine this fundamental technique and compare its performance with our new algorithms. We mention that **ColG** instantiated in the air marshal scheduling setting is precisely the ASPEN algorithm – the leading algorithm for scheduling air marshals at scale [Jain *et al.*, 2010]. In other spatio-temporal security games, though some algorithms utilize compact representation and linear programming [Fang *et al.*, 2013; Yin *et al.*, 2015], they usually *only* output an implementable marginal vector, whose decomposition into a deployable mixed strategy still requires strategy generation. To avoid this, we directly use **ColG** in these settings.

If the attacker cannot monitor any target, then all three algorithms achieve the *same* solution quality since they all implement the optimal marginal vector. So, our goal is to compare how robust these algorithms are in the presence of partial attacker observations. All algorithms are thoroughly tested in two important security settings: the *Wildlife Protection setting* (WildProt) and *Air Marshal Scheduling for round-trip flights* (AMS). Due to space limit, here we only present results that convey interesting and important messages.

In all WildProt games, the defender has two patrollers, and **MaxEn** is implemented using our algorithm in Section 4.1. All algorithms run very efficiently (games with 200 targets can be solved within a minute), so we only compare their solution qualities using the measure of attacker utilities and TDM. Whenever the attacker utility is compared, we add a **BaseLine** which is the attacker utility assuming the attacker *cannot* monitor any target. This is the best (i.e., smallest) possible attacker utility. The closer to this lower bound, the better. In all figures, the smaller the value is, the better. All results are averaged over 20 randomly generated *zero-sum* games with utility drawn from  $[-10, 10]$ . Unless specifically mentioned, we always consider that the attacker can monitor two randomly chosen targets at time  $t = 1$  and attacks one target at the last layer  $T$ . Figure 3 shows results for WildProt setting; Figure 4 contains results for AMS games.

In all these experiments, **MaxEn** significantly outperforms the classic algorithm **ColG**; **CARD** is usually slightly worse than **MaxEn**. We therefore mainly focus on analyzing other interesting findings here. Figure 3(a) and 3(b) compare the algorithms in terms of the attacker utility and TDM, respectively, by varying the number of time layers  $T$ . One interesting phenomenon uncovered from Figure 3(a) is that, as  $T$  increases, **MaxEn** and **CARD** approaches the **BaseLine**, i.e., the lowest possible attacker utility. This shows that in patrolling strategies of large entropy, the correlation between a patroller’s initial and later moves gradually disappear as time goes on. Therefore, that the attacker observes targets at the initial time layer infers almost nothing about a relatively later time layer, e.g.,  $T = 8$  is Figure 3(a). This mitigates the concerns raised at the beginning of this paper. In Figure 3(c) and 3(d), we further examine this phenomenon by fixing  $T = 9$ , and compare the algorithms by varying the number of Monitored Targets (#MoT). Surprisingly, we find that even when the attacker can monitor 6 out of 9 targets at  $t = 1$ , **MaxEn** and **CARD** are still close to **BaseLine** at  $T = 9$ , while the performance of **ColG** gradually decrease as #MoT increases.

Figure 4 compares all algorithms by varying the deployment-to-saturation (DtS) [Jain *et al.*, 2012], which is  $2k/n$  in AMS setting. DtS captures the fraction of targets that can be covered in a pure strategy. Interestingly, it turns out that the higher the DtS ratio is, the worse **ColG** performs. This is evidenced by the gap between **ColG** and **MaxEn** in Figure 4. This is because, with higher DtS, **ColG** quickly converges to an optimal mixed strategy with very small support, since each pure strategy covers many targets. Unfortunately, such a small-support strategy suffers severely from the curse of correlation. We recorded that in games with 100 targets, **MaxEn**, **CARD**, **ColG** use 99997, 2199, 53 pure strategies in average (**MaxEn** samples 100,000 pure strategies, and almost all of them are different). We remark that the trend of the gap in Figure 4(a) is similar to the trend of the gap in Figure 4(b) (also happens in Figure 3(a) 3(b)), this provides evidence that our notion of TDM correctly captures loss of unpredictability, and ultimately, of the defender utility.

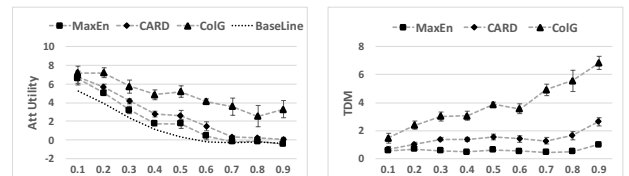


Figure 4: AMS Setting with increasing DtS Ratio ( $x$ -axis)



## References

- [Agmon *et al.*, AAMAS 2008] Noa Agmon, Vladimir Sadov, Gal A. Kaminka, and Sarit Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. volume 1, pages 55–62, AAMAS - 2008.
- [Agmon *et al.*, ICRA 2008] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. Multi-robot perimeter patrol in adversarial settings. ICRA-2008.
- [Basilico *et al.*, 2009a] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. AAMAS, 2009.
- [Basilico *et al.*, 2009b] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. WI-IAT '09. IEEE Computer Society, 2009.
- [Bernardo and Smith, 2001] José M Bernardo and Adrian FM Smith. Bayesian theory, 2001.
- [Blum *et al.*, 2014] Avrim Blum, Nika Haghtalab, and Ariel D Procaccia. Learning optimal commitment to overcome insecurity. In *NIPS*, 2014.
- [Bosansky *et al.*, 2015] B. Bosansky, Albert X. Jiang, M. Tambe, and C. Kiekintveld. Combining compact representation and incremental generation in large games with sequential strategies. In *AAAI*, 2015.
- [Bošanský *et al.*, 2011] Branislav Bošanský, Viliam Lisý, Michal Jakob, and Michal Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *AAMAS. IFAAMAS*, 2011.
- [Colbourn *et al.*, 1995] J C. Colbourn, J S. Provan, and D. Vertigan. The complexity of computing the tutte polynomial on transversal matroids. *Combinatorica*, 1995.
- [Fang *et al.*, 2013] Fei Fang, Albert Xin Jiang, and Milind Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *AAMAS*, pages 957–964, 2013.
- [Fang *et al.*, 2016] F. Fang, T. H Nguyen, R. Pickles, W. Y Lam, G. R Clements, B. An, A. Singh, M. Tambe, and A. Lemieux. Deploying paws: Field optimization of the protection assistant for wildlife security. In *IAAI*, 2016.
- [Jain *et al.*, 2010] Manish Jain, Erim Kardeş, Christopher Kiekintveld, Milind Tambe, and Fernando Ordóñez. Security games with arbitrary schedules: a branch and price approach. In *AAAI*, pages 792–797, 2010.
- [Jain *et al.*, 2012] Manish Jain, Kevin Leyton-Brown, and Milind Tambe. The deployment-to-saturation ratio in security games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1362–1370. AAAI Press, 2012.
- [Jaynes, 1957] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.
- [Kiekintveld *et al.*, 2009] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.
- [Mail, Nov 11 2016] Daily Mail. Aisle seat, they won’t nap and they’re not dressed for the weather: Pilots and cabin crew reveal how to spot an air marshal. Nov 11, 2016.
- [Moreto, 2013] William Moreto. *To conserve and protect: Examining law enforcement ranger culture and operations in Queen Elizabeth National Park, Uganda*. PhD thesis, Rutgers University-Graduate School-Newark, 2013.
- [Nguyen *et al.*, 2014] Thanh Hong Nguyen, Albert Xin Jiang, and Milind Tambe. Stop the compartmentalization: Unified robust algorithms for handling uncertainties in security games. In *AAMAS*, 2014.
- [Nyirenda and Chomba, 2012] Vincent R Nyirenda and Chansa Chomba. Field foot patrol effectiveness in kafue national park, zambia. *Journal of Ecology and the Natural Environment*, 4(6):163–172, 2012.
- [Singh and Vishnoi, 2014] Mohit Singh and Nisheeth K Vishnoi. Entropy, optimization and counting. In *STOC*, pages 50–59. ACM, 2014.
- [Tambe, 2011] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.
- [Toda, 1991] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
- [Tsai *et al.*, 2010] Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*, 2010.
- [Valiant, 1979] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [Vorobeychik *et al.*, 2014] Y. Vorobeychik, B. An, M. Tambe, and S. Singh. Computing solutions in infinite-horizon discounted adversarial patrolling game. In *ICAPS, June 2014*, 2014.
- [Xu *et al.*, 2015] Haifeng Xu, Albert X. Jiang, Arunesh Sinha, Zinovi Rabinovich, Shaddin Dughmi, and Milind Tambe. Security games with information leakage: Modeling and computation. In *IJCAI*, 2015.
- [Xu, 2016] Haifeng Xu. The mysteries of security games: Equilibrium computation becomes combinatorial algorithm design. In *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM, 2016.
- [Yin *et al.*, 2015] Yue Yin, Haifeng Xu, Jiarui Gain, Bo An, and Albert Xin Jiang. Computing optimal mixed strategies for security games with dynamic payoffs. In *IJCAI*, 2015.

## APPENDIX

### Proof of Theorem 3.1

Let binary vector  $v_A, v_B$  denote a generic realization of  $X_A$  and  $X_B$  respectively. When  $B \subset A$ , we say “ $v_A$  agrees with  $v_B$ ”, notated as  $v_A \rightarrow v_B$ , if  $v_A$  agrees with  $v_B$  at all the entries in set  $B$ . Notice that given  $v_B$ , there may be multiple  $v_A$ ’s that agree with  $v_B$  since  $B$  is a strict subset of  $A$ . Moreover,  $\forall v_B \in \{0, 1\}^B$ , we have

$$\sum_{v_A: v_A \rightarrow v_B} x_T(v_A) \cdot \Pr(X_A = v_A) = x_T(v_B) \cdot \Pr(X_B = v_B) \quad (2)$$

It is easy to verify that the  $\text{TDM}(X_T|X_A)$ , as a function of  $x_T(v_A)$  for all  $v_A$ , is convex. Using Equality (2), the convexity implies that  $\text{TDM}(X_T|X_A) \geq \text{TDM}(X_T|X_B)$ .

Consider  $\text{TDM}(X_{-i}|X_i) \geq 0$ . The “=” holds only when  $x_{-i} = x_{-i}(v_i)$  for any  $v_i$  in the support of  $X_i$ . This implies that target  $i$  is trivial –  $i$  is either full protected or fully unprotected. Otherwise, there must exist some  $j \neq i$  such that the marginal probability of  $j$  will be different between the circumstances that  $i$  is protected and not protected. This proves the theorem.

### Proof of Theorem 4.1

When there are two time steps, the game structure corresponds to a bipartite graph (recall Figure 2). It is important to notice that a pure strategy here does *not* simply correspond to a bipartite matching of size  $k$ , therefore we cannot reduce from the problem of counting size- $k$  matchings. This is because the selected  $k$  edges are allowed to share nodes. Moreover, our definition of a pure strategy is the set of covered targets, while not the edges themselves. In fact, sometimes one pure strategy can be achieved by different set of  $k$  edges.

To prove the theorem, we reduce from the problem of counting bases of a transversal matroid, which is known to be #P-complete [Colbourn *et al.*, 1995]. Given any bipartite graph  $G = (L \cup R, E)$  with  $|L| = k$ ,  $|R| = n$  and  $k \leq n$ , any set  $T \subseteq R$  is an independent set of the transversal matroid  $\mathcal{M}(G)$  of  $G$  if there exists a matching of size  $|T|$  in the subgraph induced by  $L \cup T$ ; Such a  $T$  is a base if  $|T| = |L| = k$ .

Given any bipartite graph  $G = (L \cup R, E)$  with  $k = |L| \leq |R| = n$ , we reduce counting bases of the transversal matroid  $\mathcal{M}(G)$  to the max-entropy implementation for the two-time-layer spatio-temporal security games on graph  $G$ .<sup>5</sup> Let  $\mathcal{S}_{2k}$  denote the set of pure strategies that cover exactly  $2k$  nodes. We first reduce counting bases of  $\mathcal{M}(G)$  to counting  $\mathcal{S}_{2k}$ . This simply because if a pure strategy covers  $2k$  nodes, it must cover all  $k$  nodes in  $L$  and another  $k$  nodes in  $R$ , and these  $2k$  nodes are matchable. It is not hard to see that elements in  $\mathcal{S}_{2k}$  and  $\mathcal{M}(G)$  are in one-to-one correspondence.

<sup>5</sup>Though the definition of spatio-temporal security games requires that each time layer has the same number of nodes, this requirement is not important since one can always add *isolated* nodes to each time layer to equalize the number of nodes.

Since counting reduces to generalized counting, we finally reduce generalized counting over set  $\mathcal{S}_{2k}$  to max-entropy implementation of the following special subset of marginal vectors  $\mathcal{X}_{2k} = \{\mathbf{x} \in [0, 1]^{k+n} : x_{1,i} = 1, \forall i \in L; \sum_{i=1}^n x_{2,i} = k\}$ . It is easy to see that any mixed strategy that matches a marginal vector  $\mathbf{x} \in \mathcal{X}_{2k}$  must support on  $\mathcal{S}_{2k}$ . Therefore, when considering max-entropy implementation of any  $\mathbf{x} \in \mathcal{X}_{2k}$ , we can w.l.o.g. restrict the set of pure strategies to be  $\mathcal{S}_{2k}$ . By Lemma 2.2, generalized counting over  $\mathcal{S}_{2k}$  reduce to max-entropy implementation for any  $\mathbf{x} \in \mathcal{X}_{2k}$ .

### Proof of Theorem 4.2

By Lemma 2.2, we only need to show that the generalized counting version of the problem admits a polynomial time algorithm. We now exhibit such an algorithm. For ease of presentation, our description focuses on the case with two patrollers, though it is not hard to see that the algorithm generalizes to a constant number of patrollers.

Let  $G = (V, E)$  denote the grid graph corresponding to the spatio-temporal security game. We propose a dynamic program (DP) that exploits the natural chronological order of the targets along the temporal dimension. In particular, given any set of weights  $\{\alpha_{t,i}\}$  for the generalized counting problem, for any  $1 \leq i \leq j \leq N$ , we use  $\text{DP}(i, j; t)$  to denote the solution to the counting problem restricted to the truncated graph with only time layers  $1, 2, \dots, t$ , and moreover, satisfying that the two patroller must end at cell  $i, j$  at time  $t$ . Observe that  $\text{DP}(i, j; 1) = \alpha_{1,i}\alpha_{1,j}$  when  $i \neq j$  and  $\text{DP}(i, i; 1) = \alpha_{1,i}$ . We remark that our algorithm description in the main section intentionally “omitted” the trivial conner case of  $i = j$  for ease of presentation. We then use the following update rule for  $t \geq 2$ :  $\text{DP}(i, j; t) =$

$$\begin{cases} \alpha_{t,i}\alpha_{t,j} \cdot \sum_{(i',j') \in \text{pre}(i,j)} \text{DP}(i', j'; t-1) & \text{if } i < j \\ \alpha_{t,i} \cdot \sum_{(i',j') \in \text{pre}(i,j)} \text{DP}(i', j'; t-1) & \text{if } i = j \end{cases}$$

where  $\text{pre} = \{(i', j') : i' \leq j' \text{ s.t. } v_{t-1,i'}, v_{t-1,j'} \text{ can reach } v_{t,i}, v_{t,j}\}$  is essentially the set of all pairs of nodes that can reach  $v_{t,i}, v_{t,j}$ . Note that the solution to the generalized counting problem is  $\sum_{i \leq j} \text{DP}(i, j; T)$ . The correctness of the algorithm follows from the observation that if the two patrollers are at  $v_{t,i}$  and  $v_{t,j}$ , they must come from  $v_{t-1,i'}$  and  $v_{t-1,j'}$  for certain  $(i', j') \in \text{pre}(i, j)$ . The updating rule simply aggregates all such choices. The algorithm runs in  $\text{poly}(N, T)$  time.

### Proof of Theorem 4.3

Let  $G = (A \cup B, E)$  denote the bipartite graph for the air marshal scheduling problem,  $|A| = n_1, |B| = n_2$ . We first observe that  $G$  is a union of multiple isolated smaller bipartite graphs, each containing all flights between  $a$  and another city (see Figure 5). This is because any flight from  $a$  to city  $b$  can never form a round trip with a flight from another city  $c$  to  $a$ . We will call each isolated bipartite graph a *component*.



Figure 5 has two components. Within each component, we sort the flights in  $A$  by their *arrival* time and flights in  $B$  by their *departure* time.

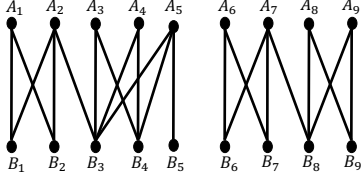


Figure 5: Consistent round-trip flights between a domestic city and two outside cities.

We now show that the generalized counting over the set of defender pure strategies admits a polynomial time algorithm. By Lemma 2.2, this proves the theorem.

Since each flight has at most one air marshal, any assignment of air marshals must correspond to a matching in  $G$ . However, a pure strategy  $S$  – i.e., a set of covered flights – can be accomplished by different matchings. For example, the set  $S = \{A_1, A_2, B_1, B_2\}$  in Figure 5 can be achieved by the matching  $\{(A_1, B_1), (A_2, B_2)\}$  or the matching  $\{(A_1, B_2), (A_2, B_1)\}$ . Nevertheless, only the matching  $\{(A_1, B_1), (A_2, B_2)\}$  is ordered (see Definition 4.4). We first prove Lemma 4.5 which states that pure strategies and *ordered*  $k$ -matchings are in one-to-one correspondence.

*Proof of Lemma 4.5.* It is easy to see that any ordered  $k$ -matchings corresponds to one pure strategy. We prove the reverse. Given any pure strategy  $S$  consisting of  $2k$  flights, let  $\tilde{E} = \{e_1, \dots, e_k\}$  be any matching that results in  $S$ . We claim that if there exist two edges  $e, e' \in \tilde{E}$  with  $e = (A_i, B_j)$  and  $e' = (A_{i'}, B_{j'})$  such that  $i > i'$  and  $j < j'$ , then  $(A_i, B_{j'})$  and  $(A_{i'}, B_j)$  must also be edges in  $E$ . Since  $e, e' \in \tilde{E}$ , we must have  $T_1 < \text{depart}(B_j) - \text{arrival}(A_i) < T_2$  and  $T_1 < \text{depart}(B_{j'}) - \text{arrival}(A_{i'}) < T_2$ . Since flights in  $A$  are ordered increasingly by arrival time and flights in  $B$  are ordered increasingly by departure time, we have  $\text{arrival}(A_i) \geq \text{arrival}(A_{i'})$  and  $\text{depart}(B_j) \leq \text{depart}(B_{j'})$ . These inequalities imply  $\text{depart}(B_{j'}) - \text{arrival}(A_i) \leq \text{depart}(B_{j'}) - \text{arrival}(A_{i'}) \leq T_2$  and  $\text{depart}(B_{j'}) - \text{arrival}(A_i) \geq \text{depart}(B_j) - \text{arrival}(A_i) \geq T_1$ , therefore  $(A_i, B_{j'}) \in E$ . Similarly, one can show that  $(A_{i'}, B_j) \in E$ .

As a result, we can adjust the matching by using the edges  $(A_i, B_{j'})$  and  $(A_{i'}, B_j)$  instead. Such adjustment can continue until the matching becomes ordered. The procedure will terminate within a finite time by a simple potential function argument, with potential function  $f(\tilde{E}) = \sum_{e=(A_i, B_j) \in \tilde{E}} |i - j|^2$ . In particular, the above adjustment always strictly decreases the potential function since  $|i - j|^2 + |i' - j'|^2 > |i - j'|^2 + |i' - j|^2$  if  $i > i'$  and  $j < j'$ . The adjustment will terminate with an ordered matching. It is not hard to see that the ordered matching is unique. This concludes our proof.  $\square$

Lemma 4.5 provides a way to reduce generalized counting over the set of pure strategies to generalized counting of size- $k$  ordered matchings. In particular, given any set of non-negative weights  $\alpha \in \mathbb{R}_+^{n_1+n_2}$ , we define edge weight  $w_e = \alpha_{A_i} \alpha_{B_j}$  for any  $e = (A_i, B_j) \in E$ . Therefore, the weight of any pure strategy equals the weight of the corresponding size- $k$  ordered matching with edge weights  $w_e$ 's. The following lemma shows that generalized counting of size- $k$  ordered matchings admits an efficient algorithm.

**Lemma 6.1.** *Given a bipartite graph  $G = (A \cup B, E)$  with non-negative edge weights  $w_e$  for  $e \in E$ , the problem of counting size- $k$  ordered matchings admits a  $\text{poly}(n, k)$  time algorithm where  $n = |A \cup B|$ .*

*Proof.* Throughout the proof, we will always use symbol  $M$  to denote an *ordered* matching and  $|M| = d$  means that the matching has size  $d$ . For convenience we denote  $w_M = \prod_{e \in M} w_e$ . Define  $E_{l,r} \subseteq E$  to be the set of edges that are “under”  $A_l$  and  $B_r$ , where  $A_l \in A, B_r \in B$ . Formally, any  $e = (A_i, B_j) \in E$  is in  $E_{l,r}$  iff  $i \leq l, j \leq r$ .

We build the following dynamic programming table  $\text{DP}(l, r; d) = \sum_{M: M \subseteq E_{l,r}, |M|=d} w_M$ . In other words,  $\text{DP}(l, r; d)$  is the sum of the weight of all size- $d$  ordered matchings with edges in  $E_{l,r}$ . Algorithm 2 gives a dynamic program that computes the DP table. The correctness of the algorithm follows almost by definition. In particular, to compute  $\text{DP}(l, r; d)$ , we enumerate all the possibilities of the uppermost edge in the matching, which must be one of the following two cases: either the uppermost edge is an edge  $(A_i, B_j)$  with  $i = l$  or  $j = r$ , or does not cover  $A_l$  and  $B_j$ . The former case multiplies the weight of the uppermost edge with the recursively computed  $\text{DP}(i-1, j-1; d-1)$ , while the later case degenerates to  $\text{DP}(l-1, r-1; d)$ , as reflected in the recursion formula. It is easy to check that the running time of the algorithm is  $\text{poly}(n, k)$ .  $\square$

---

**Algorithm 2** Generalized Counting of Ordered  $k$ -Matchings

---

**Input:**  $G = (A \cup B, E)$ ;  $w_e \geq 0$  for any  $e \in E$ .

**Output:**  $\sum_{M: |M|=d} w_M$  %  $M$  is an ordered matching

- 1: **Initialization:**  $\text{DP}(l, r; 0) = 1$  for  $l = 0, \dots, n_1, r = 0, \dots, n_2$ ;  $\text{DP}(0, r; d) = \text{DP}(l, 0; d) = 0$  for all  $d \geq 1, l = 0, 1, \dots, n_1, r = 0, 1, \dots, n_2$ .
- 2: **Update:** for  $d = 1, \dots, k, l = 2, \dots, n_1, r = 2, \dots, n_2$ :

$$\text{DP}(l, r; d) = T(l-1, r-1; d) + \sum_{\substack{e=(A_i, B_j) \in E_{l,r} \\ \text{s.t. } i=l \text{ or } j=r}} w_e \cdot \text{DP}(i-1, j-1; d-1).$$

- 3: **return**  $\text{DP}(n_1, n_2; k)$ .
-